Minnowbrook '94

Formal Methods Demonstration Project for Space Applications

July 27, 1994

Rick Covington, Ph.D. and John C. Kelly, Ph.D.
Software Product Assurance Section
Jet Propulsion Laboratory
Pasadena, CA

Contents

- Problem Statement
- What are Formal Methods
- Overview of NASA Code Q Research Proposal
- Application: Shuttle Jet Select
- Findings from Jet Select Application
- Other Applications in Progress

Introduction: Current Problems in Engineering Software for Critical Subsystems

Requirements and design specifications are a high priority candidate for better software engineering techniques

- Most hazardous software safety errors found during system integration and test of two NASA spacecraft were the result of requirements discrepancies or interface specifications [Lutz93].
- The highest density of major defects found through the use of software inspections was during the requirements phase. This was 7 times higher than the density of major defects found in code inspections [Kelly92].
- Requirements errors are between 10 and 100 times more costly to fix at later phases of the software lifecycle than at the requirements phase itself [Basili84], [Boehm84], [Kelly92].
- One study found that early lifecycle errors are the most likely to lead to catastrophic failures [Leve86].

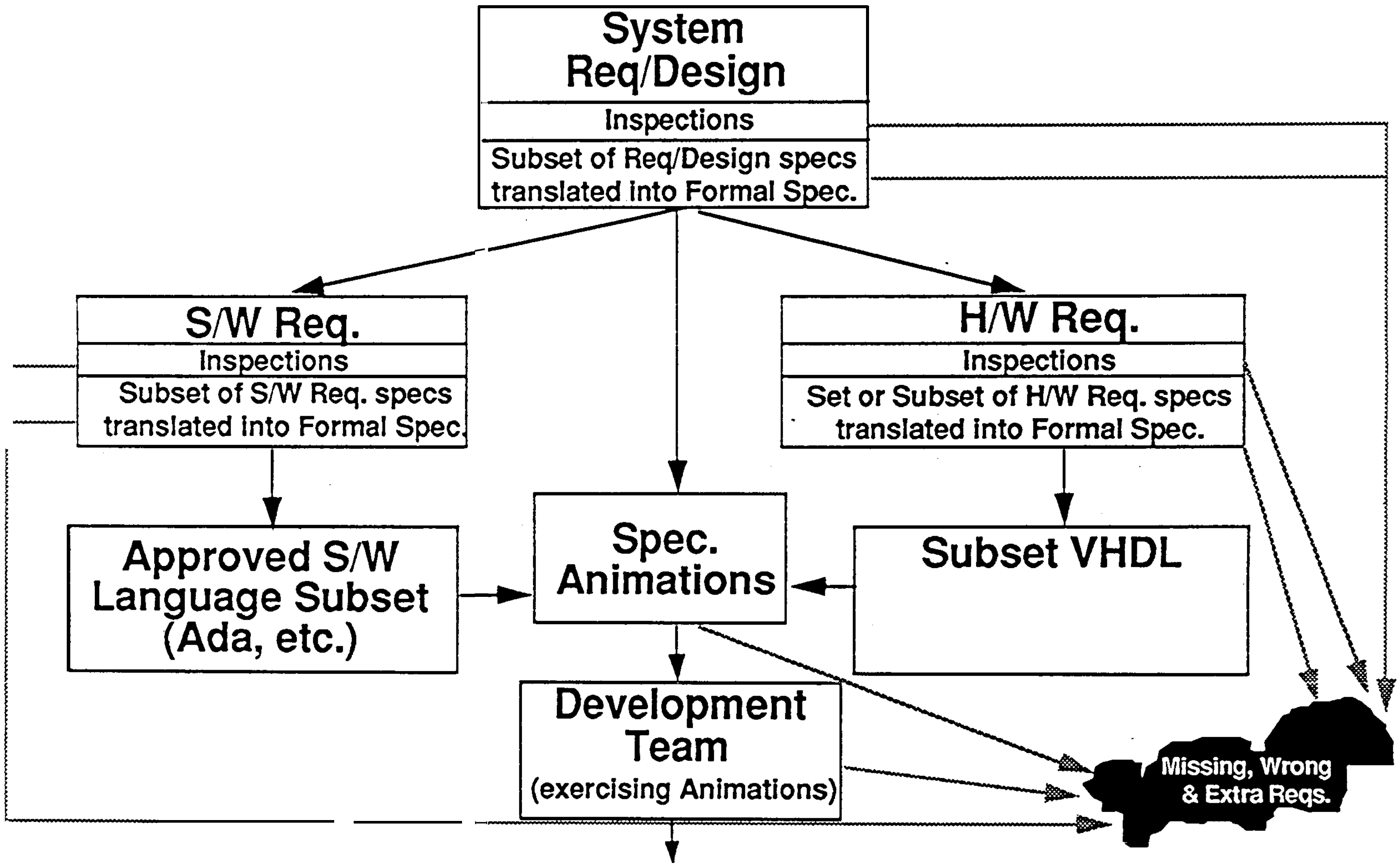
Introduction: What are Formal Methods?

- Formal Methods refer to the use of techniques and took based on formal **logic** and mathematics used to specify and verify systems, software, and hardware.
- Provide a precise “abstract?’ mathematical model of a component’s specification
- Complement empirical methods such as traditional testing
- At the most rigorous level Formal Methods benefit from power of automated deductive reasoning which can be used to formally prove logical assertions about a **system**

Introduction (continued)

- Why use Formal Methods?
  - Increasing concern about the use of complex software in life-critical and mission-critical applications
  - The high cost of testing and fixing problems late in the development process
  - Improvements in FM techniques and tools over the last 10 years
- Who's using Formal Methods?
  - FM are being used in many critical applications:
    - Secure Networks and Operating Systems
    - Nuclear Reactor Shutdown procedures
    - Automated Train Controllers
    - Air Traffic Collision Avoidance Systems
    - Active European use, including Draft Standards

Formal Methods can be used for System, Software, and Hardware Specifications



Purpose

- The Goal of this study is to demonstrate the applicability of Formal Methods techniques on critical NASA software subsystems

  Phase I Task: Demonstrate the Applicability of Formal Methods to Shuttle's On-Board Jet Select Software Subsystem (A highly critical, yet relatively stable set of requirements)

- Phase II Tasks: Demonstrate Formal Methods on several smaller projects which are currently developing critical software and provide guidance at the managerial level

  Phase III Tasks: Demonstrate Formal Methods on a large critical project in the early development stages and provide guidance at the technical level

Introduction: Team Members

- Jet Propulsion Laboratory
  - John Kelly, Ph.D., Rick Covington, Ph.D., Robyn Lutz, Ph.D., Al Nikora, Brent Auernheimer, Ph.D. (CSUF), Yoko Ampo (NEC), Ken Abernethy, Ph.D. (FU)
- Johnson Space Center
  - Ernie Fridge, David Hamilton (LORAL), Mike Beims (LORAL-Shuttle RA), Chris Hickey (LORAL-Shuttle RA),
- Langley Research Center
  - Rick Butler, Ben DiVito, Ph.D. (VIGYAN), John Rushby, Ph.D. (SRI), Judith Crow, Ph.D. (SRI), Sam Owre (SRI)
- NASA HQ Sponsor: Alice Robinson
- Alumni
  - Betty Cheng, Ph.D. (MSU), Mori Khorrami (JPL), Doc Shankar, Ph.D. (IBM), Scott French (LORAL), Sally Johnson (LaRC)
- Advisors
  - Susan Gerhart, Ph.D. (UHCL) & Charles Hardwick, Ph.D. (UHCL)

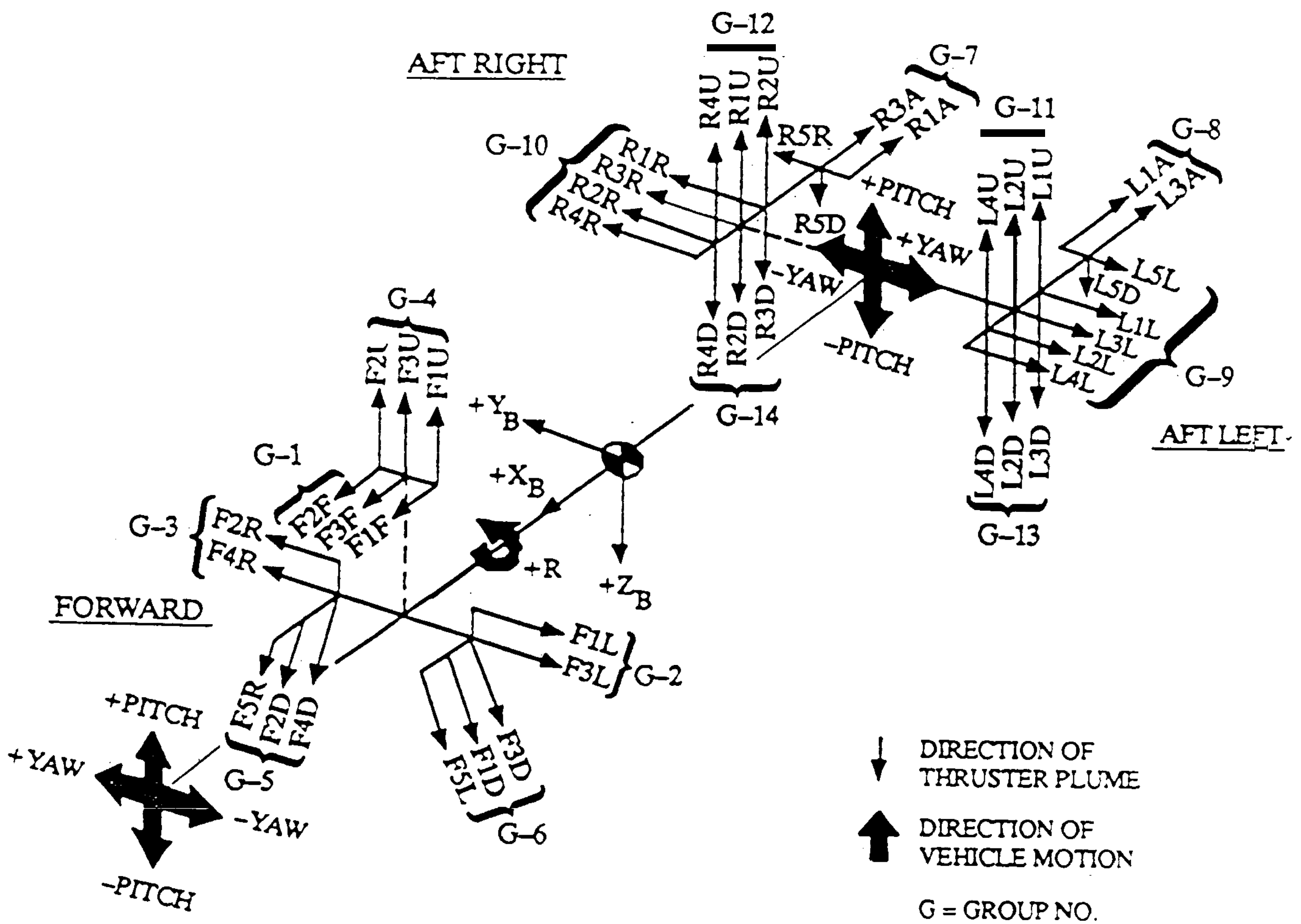
The Prototype Verification System (PVS)

- An integrated environment for the development and analysis of **formal** specifications
- Supports a wide range of activities involved in creating, analyzing, modifying, managing, and documenting formal specifications
- PVS consists of:
  - Specification language, a parser, a typechecker, a prover, a **prettyprinter,** specification **libraries,** various browsing tools, **syntax similar** to Ada, **all** integrated through a GNU **Emacs** interface
- Developed at SRI, International in Menlo Park, CA

Planned Technical Approach

- Step O: Task Preparation
- Step 1: Formal Methods Startup Exercise
- Step 2: Formal Model, Specification, & Animation for Jet Select
- Step 3: Formulation & Proof of Properties

The Shuttle's Control Jets



Shuttle's Jet Select Formal Methods Products

- Three levels of specifications converted using Formal Methods (PVS)
- Ada Emulator of Vernier/ALT Jets
- Proofs of High Level Properties
- Issues List
- Case Study Report

Sample of the Current Functional Subsystem Software Requirements (FSSR) Document

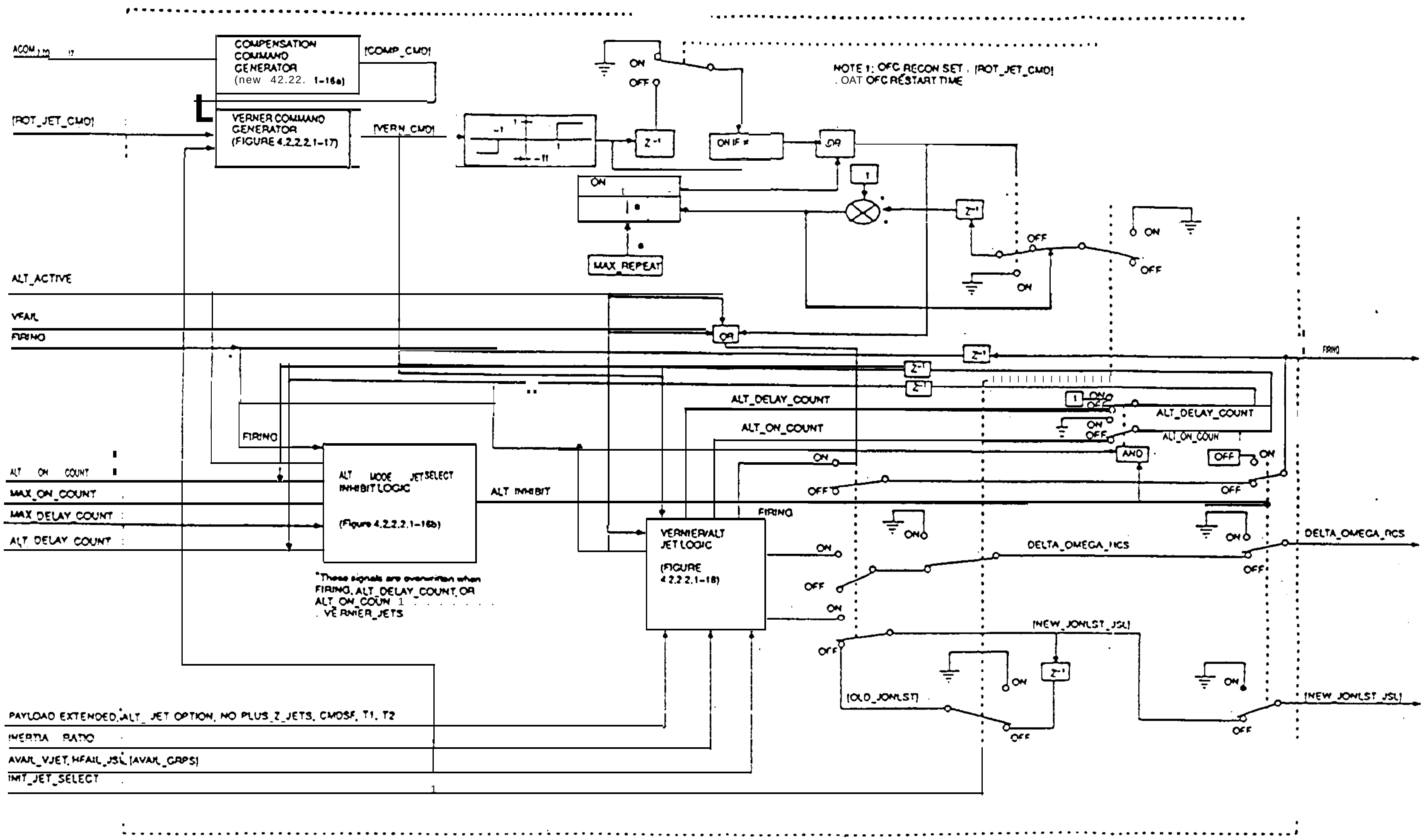
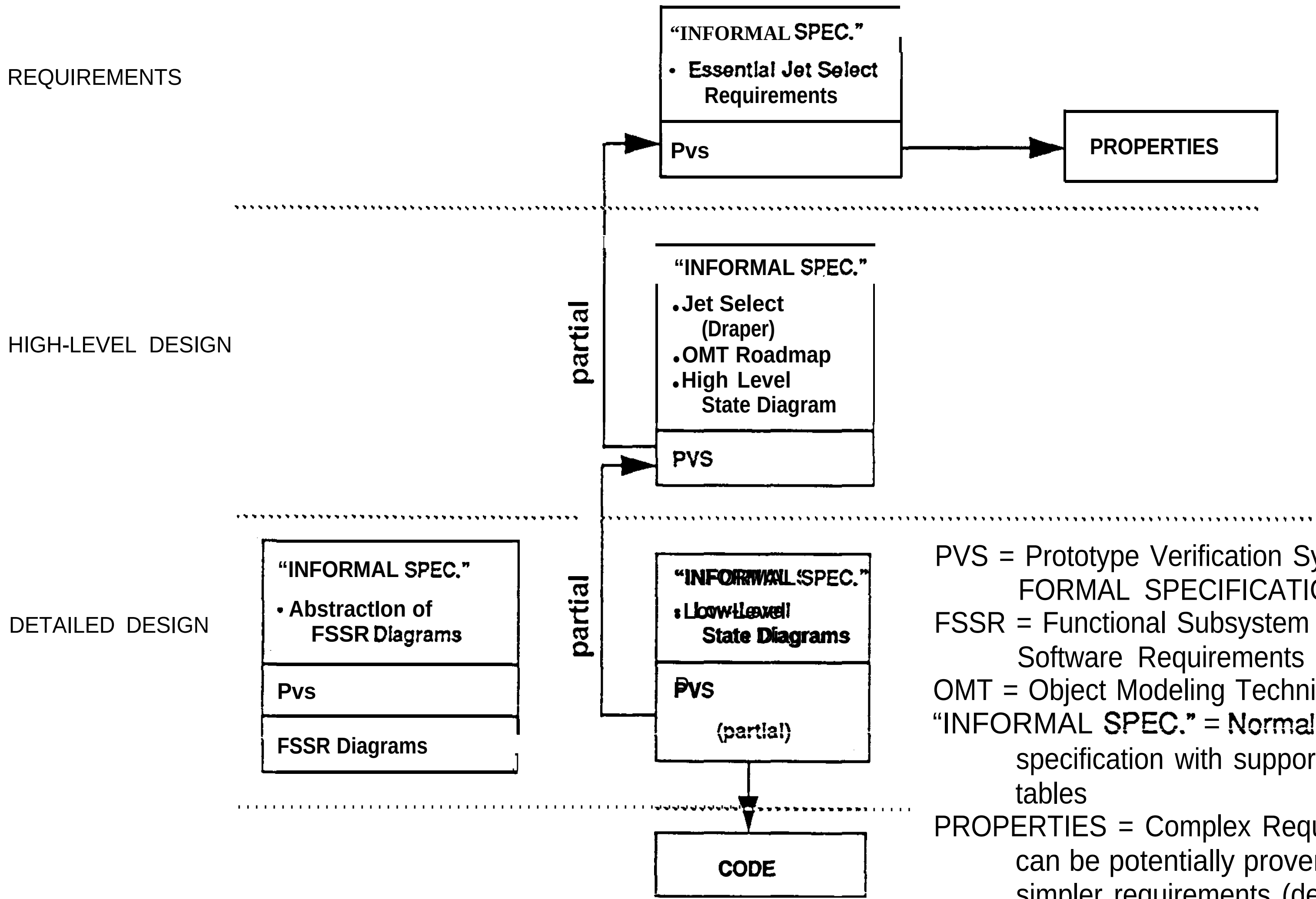


Figure 4,2,2,2, 1-16. VERNIER_ALT_JETS

Levels of Specifications



PVS = Prototype Verification System (the FORMAL SPECIFICATION)

FSSR = Functional Subsystem Software Requirements

OMT = Object Modeling Technique

“INFORMAL SPEC.” = Normal English specification with supporting figures and tables

PROPERTIES = Complex Requirements which can be potentially proven from simpler requirements (definitions & axioms)

Detailed FSSR Diagram

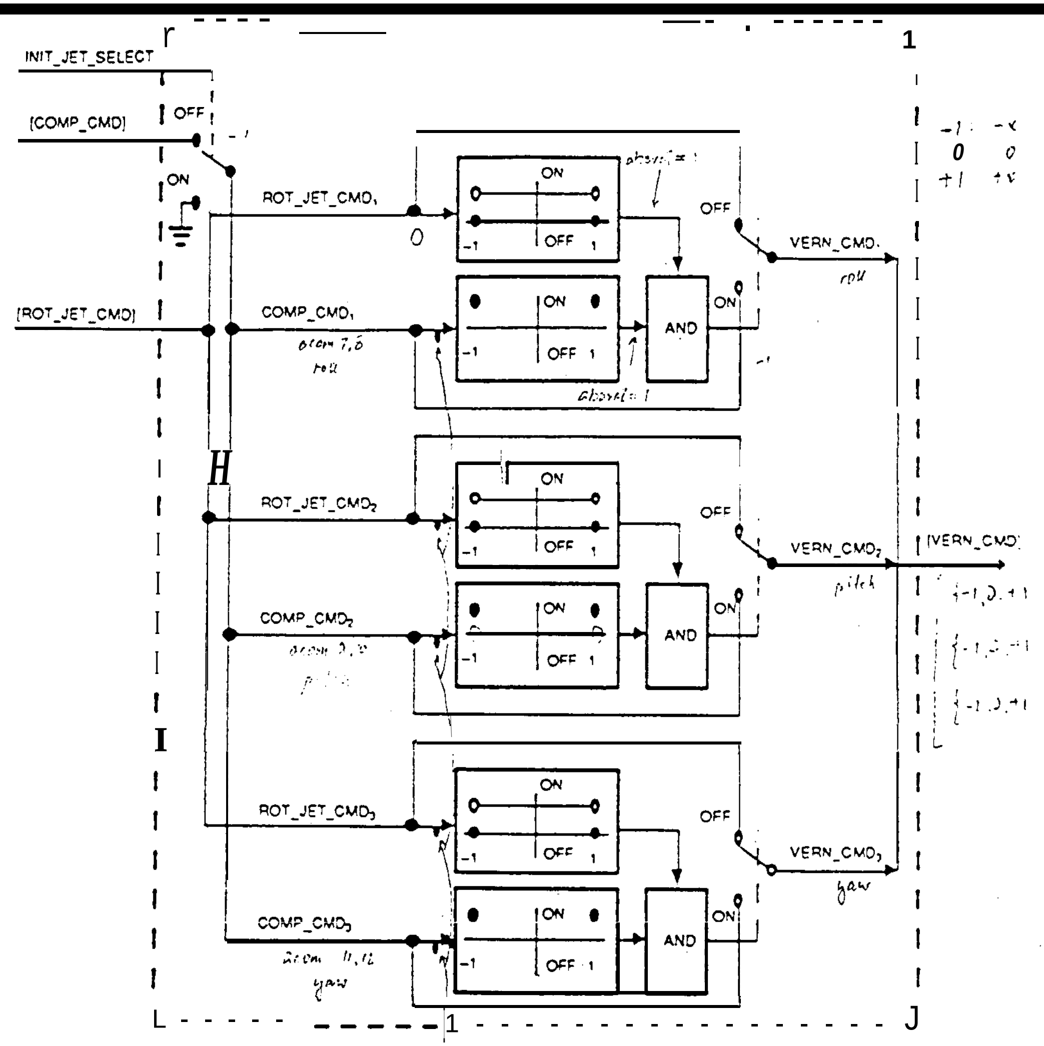


Figure 4.2.2/! .1-17. Vernier Command Generator

Example of a Formal Specification for the FSSR Diagram

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                %
%        LOCAL FUNCTION SPECIFICATIONS           %
%                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
    box1_rot_fnc:[real -> boolean] = (LAMBDA (x:real) :
                                        x > -1 AND x < 1)
%
%
%
    box2_comp_fnc:[real -> boolean] = (LAMBDA (x:real) :
                                        x = -1 OR x = 1)
%
%
%
    ord : [rotation ->  nat = (LAMBDA (r:rotation) :
       CASES r OF
         roll  : 1,
         pitch : 2,
         yaw   : 3
       ENDCASES)
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                %
%        MAIN FUNCTION SPECIFICATIONS            %
%                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
vernier_cmd_gen_logic (init_jet_select, comp_cmd, rot_jet_cmd):
                        scalar_rotation_direction =
%
   (LAMBDA  r:rotation)    IF init_jet_select
                              THEN rot_jet_cmd(r)
                           ELSE IF box1_rot_fnc(rot_jet_cmd(r)   AND
                                   box2_comp_fnc(comp_cmd(r))
                                     THEN comp_cmd(r)
                                ELSE rot_jet_cmd(r)
                                ENDIF
                           ENDIF)
%
%
%
END vernier_command_generator
%
%
```

Critical Properties Considered for Proof of Consistency using PVS (at High-Level Req.)

- Jet Select shall provide multiple algorithms for choosing jets and allow the choice of which algorithm to use. - ***Specifications insufficient to prove***
- Jet Select shall always choose a jet if there exists an available jet that satisfies the constraints. - ***Proved***
- If ALT mode is active, Jet Select shall choose only primary jets.- ***Proved***
- When choosing primary jets, Jet Select shall choose the highest priority available jet in a given group. The priorities of each jet within a group will be predefined.- ***Proof deferred until lower level functions are defined***
- In ALT mode, Jet Select shall never choose more than 3 jets.- ***Proved***
- If Vernier mode is active, Jet Select shall choose only vernier jets. - ***Proved***
- In low +z mode, Jet Select shall not choose any jet that fires primarily in the +z direction.- ***Proved weaker lemma***
- In tail-only mode and not in low +z mode, Jet Select shall choose only jets in an aft group. - ***Proved***

Some Results and Lessons Learned

- **Current Working Specifications (“Wiring Diagrams”)**
  - **They strongly suggest specific implementation** (“How” vs. “What/Why”)
  - They are unnecessarily complex
  - Their detail sometimes obscures simple underlying function
  - They make it difficult to predict effect of modifications
- **Formal Specifications**

  Discovered issues in a mature requirements specification
  - Helped to discover “true” underlying requirements
  - Eliminated idiosyncratic function notation
  - Reduced bias toward specific implementation

General Conclusions from Phase I Demonstration

- Most benefit from Formal Methods achieved when:
  - Applied to high-level requirements
  - Applied to applications that lend themselves to abstract specifications (i.e. logically complex subsystems)
- Learning the PVS Formal Specification System was not difficult
- Requirements Analysts were willing and able to understand specifications in the PVS language
- PVS language and tools sufficiently mature for representing Space Shuttle Jet Select software requirements

Projects

- Completed
  - Shuttle Jet Select
- In Progress

– Shuttle GPS CR	Ben DiVito, VIGYAN/LaRC
– Shuttle 3 Engine Out CR	Judy Crow, SRI/LaRC; David Hamilton, Loral/JSC
– Shuttle Mir Docking CR	David Hamilton
– Shuttle Orbit DAP	David Hamilton
– Cassin CDS FP S/W	Robyn Lutz, JPL; Yoko Ampo, NEC/JPL
– Station EPS	David Hamilton Rick Covington, JPL

The research described in this presentation was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.